

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Utility Patent Application

SYSTEM-WIDE SELECTIVE ACTION MANAGEMENT

Inventor(s):

Marc Shapiro

James O'Brien

Caroline Matheson

Pablo Rodriguez

Manuel Costa

CLIENT'S DOCKET NO. MS305378.1

ATTORNEY'S DOCKET NO. MS1-1730US

88436702939

SYSTEM-WIDE SELECTIVE ACTION MANAGEMENT

Related Applications

The present application is a continuation-in-part of U.S. Patent No. 10/602,201 entitled "RÉCONCILABLE AND UNDOABLE FILE SYSTEM" and filed on June 24, 2003, incorporated herein by reference for all that it discloses and teaches.

Technical Field

The invention relates generally to computer system management, and more particularly to system-wide selective action management.

Background

Computers are complicated devices that can provide rich functionality through a large variety of application and operating system (OS) features. However, computer applications and systems remain hard to use in many situations for many users. Moreover, computer applications and systems can be unforgiving when something goes wrong. As interconnections between applications and between systems increase, the opportunities for a user's small mistake to corrupt large amounts of state throughout one or more computer systems multiply.

One approach to helping the user out in such situations includes the familiar “undo” feature known in many contemporary applications. However, this approach can be inadequate at least because the effects of some “mistakes” can propagate throughout multiple applications, including the operating system, and

1 through other interconnected systems. In addition, implementations of this
2 approach tend to be strictly chronological in nature and applicable in only a small
3 time frame (e.g., for only a limited number of actions or only while the application
4 is in its current execution instance). In contrast, another approach allows
5 restoration of an OS state (e.g., after a system crash or to return to a previous OS
6 version), but this approach does not integrate its restorative effects into multiple
7 applications executing within one or more systems nor does it apply to non-OS
8 applications. Accordingly, existing approaches leave much room for
9 improvement.

10 **Summary**

11 Implementations described and claimed herein address the foregoing
12 problems by providing a system-wide selective action management tool.
13 Exemplary implementations of the tool can support selective action management
14 for multiple applications (including the operating system and its components, such
15 as a file system) executing on one or more computer systems. A system-wide
16 action management tool can log actions performed on the computer system(s) and
17 record relationships among such actions (e.g., actions associated with other
18 computer-related sources, including different documents, different applications
19 and even different computer systems). When a user discovers a mistake, the tool
20 allows the user to select one or more past actions (i.e., the “mistake”) for removal
21 or for replacement with one or more corrected actions. Given this action selection,
22 the tool can also roll back intervening actions executed between the most recent
23 action and the selected action(s). The tool can also re-execute dependent actions
24 to restore the relevant state of the system at the time of the designation, absent the
25

1 “mistake”. As such, actions throughout the system can be selectively undone,
2 fixed and/or redone in an exemplary system-wide selective action management
3 tool.

4 In some implementations, articles of manufacture are provided as computer
5 program products. One implementation of a computer program product provides a
6 computer program storage medium readable by a computer system and encoding a
7 computer program. Another implementation of a computer program product may
8 be provided in a computer data signal embodied in a carrier wave by a computing
9 system and encoding the computer program.

10 The computer program product encodes a computer program for executing
11 a computer process on a computer system. Action information that pertains to an
12 action of a first computer-related source and an associated relationship with a
13 recorded action of a second computer-related source is recorded. An action
14 management operation is executed on the action of the first computer-related
15 source and the recorded action of the second computer-related source.

16 In another implementation, a method is provided. Action information that
17 pertains to an action of a first computer-related source and an associated
18 relationship with a recorded action of a second computer-related source is
19 recorded. An action management operation is executed on the action of the first
20 computer-related source and the recorded action of the second computer-related
21 source.

22 In another implementation, a system includes an action log that records
23 action information pertaining to an action of a first computer-related source and an
24 associated relationship with a recorded action of a second computer-related
25 source. An action management module executes an action management operation

1 on the action of the first computer-related source and the recorded action of the
2 second computer-related source.

3 Other implementations are also described and recited herein.

4 **Brief Descriptions of the Drawings**

5 FIG. 1 illustrates a computer system executing an exemplary action
6 management module.

7 FIG. 2 illustrates an exemplary selective action management module
8 interacting with application-specific action management facilities.

9 FIG. 3 illustrates an exemplary hierarchy of logical actions.

10 FIG. 4 illustrates a user interface for an exemplary selective action
11 management facility.

12 FIG. 5 illustrates operations of an exemplary selective action management
13 process.

14 FIG. 6 illustrates a system useful for implementing an embodiment of the
15 present invention.

17 **Detailed Description**

18 In one implementation, a system-wide selective action management facility
19 operates on logical actions and relationships among logical actions. Logical
20 actions may be associated with different applications (including operating system
21 processes) within a computer system or distributed among multiple computer
22 systems. Generally, logical actions may represent, with varying levels of
23 granularity, operations, commands, primitives, method calls, etc. executed at the
24 application or kernel levels of a computer system. For example, a user may

1 instruct the system to change the current style (e.g., font style, paragraph style,
2 etc.). The action of “change style” is one example of a logical action. In addition,
3 the “change style” may entail multiple component actions (i.e., at a finer level of
4 granularity), such as “bold selected text”, “underline selected text”, and “change
5 font of selected text”. It should be understood that any number of levels of
6 granularity may exist, and in some implementations, the level of granularity seen
7 by the user may be managed by the user through a user interface (UI).

8 Individual actions in a system may be selected to be undone, fixed (e.g.,
9 replaced), and redone. To undo an action makes it inactive, which “rolls back” its
10 effects. In one implementation, the selected action may be undone without
11 “rolling back” *all* intervening actions (e.g., by only “rolling back” those actions
12 that depend, directly or indirectly, on the selected action). In addition, in one
13 implementation, such action management operations may also be “previewed”,
14 such that the operations are executed or rolled back as appropriate, and the results
15 are displayed, but the results are not persisted in the system(s). As such, the user
16 can view the projected results of the operations without permanently altering the
17 current system state. Accordingly, system-wide selective action management can
18 extend beyond the strictly chronological constraints of traditional undo/redo
19 systems.

20 FIG. 1 illustrates a computer system 100 executing an exemplary action
21 management module 102. A first computer-related source, such a word
22 processing application 104, and a second computer-related source, such as a
23 spreadsheet application 106, are executing on the computer system 100. It should
24 be understood, however, that the applications may be executing on different
25 computer systems, including computer systems that are separate from the

1 computer system executing the action management module 102. Furthermore,
2 while each application may be considered a computer-related source, other
3 computer-related sources may also include documents, objects, system levels (e.g.,
4 application level, kernel level), computer systems, computer processes, and other
5 action sources.

6 Each of the applications 104 and 106 may include its own traditional
7 undo/redo facility or an enhanced application-specific action management facility.
8 If so, each application 104 and 106 can use its facility to perform some level of
9 action management within the individual application. Notwithstanding, in one
10 implementation, each application 104 and 106 registers with the action
11 management module 102 in order to participate in the system-wide selective
12 action management features. Such registration requests may be received from
13 within the computer system 100, or they may be received from another computer
14 system.

15 Each application 104 and 106 records a log of actions it executes and the
16 relationships each action has with other actions. For example, the application 104
17 records a log 110 of actions performed during editing of a document (i.e.,
18 “DocA”). The log 110 records the actions (1)-(4) and their relationships to other
19 actions.

20 Two of the actions in the log 110 (i.e., actions (1) and (3)) have a
21 relationship (i.e., a predecessor-successor dependency) between them. In one
22 implementation, the relationship is recorded in association with each related
23 action, although in other embodiments, the relationship may be recorded in
24 association with only one action or the other. In addition, one action in the
25

1 log 110 (i.e., action (3)) has a relationship (i.e., also a predecessor-successor
2 dependency) with an action executed by the application 106.

3 The action management system depicted in FIG. 1 may record actions and
4 relationships that are provided externally. Several cases are possible (not depicted
5 in the figure). For example, the user might indicate explicitly her intent to relate
6 two apparently unrelated actions such that two actions designated to be part of an
7 atomic “parcel”. A relation might alternatively be part of operating system
8 semantics. For example, there is a causal (also called “predecessor-successor”)
9 relation between a copy and a paste action. Relations may also be inferred by
10 heuristics. For example, if the user reads one document and writes another
11 immediately afterwards in time, the system may infer that the latter depends on the
12 former.

13 In an exemplary implementation, the logs 110 and 112 contain a data
14 structure specifying the actions and their relationships with other actions. One
15 such exemplary data structure contains the following fields:

- 16 (1) An Action ID – identifying the primary action (e.g., a GUID (“Globally
17 Unique IDentifier”))
- 18 (2) A Source ID – identifying the source (e.g., application, document,
19 process, system level, computer system, etc.) that executed the primary
20 action (e.g., a GUID)
- 21 (3) An Action Descriptor – describing the action such that the action can be
22 undone and/or redone (e.g., “mkdir D1”)
- 23 (4) A Container Action ID – identifying a component action of which the
24 primary action is a component action

1 (5) A Component Action ID – identifying a component action that is a
2 component of the primary action (which is the corresponding container
3 action). In another implementation, container-contained relations are
4 designated as just a specific type of relationship (see Field 7 below).

(6) An “Inactive” Flag – indicating that the action is not active (e.g. has been undone), allowing a distinctive display in the UI. For example, inactive actions might be grayed out. In another implementation, an active status may be designated by a causal relationship with a special “always done” action and an inactive status may be designated by a causal relationship with a special “never done” action.

11 (7) A Relationship Descriptor – describing the relationship with another
12 action (e.g., NULL, Predecessor, Successor, Parcel)

13 (8) The Related Action ID – identifying the action to which the primary
14 action is related (e.g., a GUID)

15 (9) The Related Source ID – identifying the source that executed the related
16 action (e.g., a GUID)

17 In addition, in some implementations, fields (7)-(9) may be repeated to
18 specify relationships with different related actions or to specify different
19 relationship types with the same related action. Other data structures may be
20 employed in alternative implementations.

21 Various implementations may employ logs in different ways. For example,
22 in a simpler implementation suggested by FIG. 1, each application merely records
23 its action information (which generally includes actions and relationships) and
24 passes its action log to the action management module 102. The action
25 management module 102 then provides a user interface 108 for action

1 management operations associated with all registered sources. While feasible, this
2 approach does not leverage any (possibly existing and probably more user-
3 familiar) action management facilities of individual applications.

4 Nevertheless, given the various logs of actions and relationships, the action
5 management module 102 can provide an integrated action management UI 108
6 that displays recorded actions at a given level of granularity for all registered
7 sources and receives action selections and action management commands relating
8 to the displayed actions. For example, through the action management UI 108, a
9 user may select an action, instruct the action management module 102 to “fix” the
10 selected action, and input information for the corrected action. Fixing an action
11 may consist of undoing the actions that depend upon it, undoing the action itself,
12 replacing the action with a corrected action, executing the corrected action, and
13 redoing dependent actions. This returns the system to a corrected state. If this is
14 not the desired state, the user may perform further fixes, or revert back to the
15 original state.

16 In some circumstances, a selected action may be related to an action of a
17 computer-related source that is inactive (e.g., not currently executing or
18 accessible). In one implementation, the associated action management command
19 fails if the selected action or an intervening related action is related to an action of
20 an inactive source. In an alternative implementation, the action management
21 module 102 may execute, open or otherwise gain access to the inactive source in
22 order to access the related action (e.g., to execute the action management
23 command on the related action of the source).

24 In one implementation, actions that have been “undone” remain persisted in
25 the log, albeit marked as inactive. For example, undone inactive actions may be

1 displayed as “grayed out” in the UI 108. In this manner, inactive actions may be
2 selected for other action management operations after they have been undone.

3 FIG. 2 illustrates an exemplary selective action management module 200
4 interacting with application-specific action management facilities 202 and 204 of
5 applications 206 and 208, respectively. In the illustrated implementation,
6 individual applications can maintain their own action logs and provide action
7 management functionality through their individual action logs and their individual
8 action management UI facilities. In such implementations, information regarding
9 the cross-source relationships and the related sources may be passed to the action
10 management module 200, as shown in the record of a system-wide log 210. The
11 actions “2*” and “5*” recorded in the system-wide log 210 may be full or partial
12 instances of the actions “2” and “5” in the facilities 202 and 204 or may be
13 references to the action information managed by these facilities.

14 In one example, if the source IDs of a selected action and a related action
15 do not match, the application-specific action management facility may indicate
16 that it cannot perform the selected action management operation (e.g., undo) and,
17 therefore, refer the user to the system-wide action management facility, controlled
18 by the action management module 200.

19 In yet another alternative implementation, an application-specific action
20 management facility, such as facility 202, can detect the cross-source relationship
21 and then interact with the action management module 200 to perform the selected
22 action management operation. For example, the application-specific action
23 management UI 202 can highlight its action (2) as having a cross-source
24 relationship. Therefore, if the action (2) is selected for or implicated in an undo
25 operation, the application specific action management facility 202 notifies the

1 action management module 200 of the related action IDs, the source IDs, the
2 relationships, and the selected action management command. The action
3 management module 200 can also obtain the related action IDs, source IDs, and
4 relationships from its own logs. The action management module 200 then
5 coordinates the selected action management operation among the related sources,
6 either through the individual application-specific action management UIs, where
7 possible, or by providing its own system-wide action management UI 212, which
8 can integrate the action information from all registered sources.

9 As discussed previously, the related source need not be active at the time
10 the cross-source action management operation is selected. For example,
11 application 208 may be closed at the time action (2) of the application 206 is
12 selected for an “undo” operation. In such circumstances, the undo operation may
13 fail, or the application 206 may be started in order to complete the undo operation.

14 FIG. 3 illustrates an exemplary hierarchy of logical actions. Previous
15 descriptions herein have focused on one level of action granularity. However, in
16 some implementations, actions may be displayed and operated on at various levels
17 of granularity. As shown in FIG. 3, two container actions 300 and 302 (i.e., “edit
18 DocA” and “edit DocB”) may be displayed in one or more action management
19 UIs. Either action may be selected for a given action management operation. For
20 example, action 302 may be selected for an “undo” operation. As such, each of
21 the component actions 304, 306, 308, and 310 of the container action 302 are
22 undone (i.e., a previous system state is restored, absent the undone action 302).
23 Note that action 310 is dependent upon action 304.

24 Each component action may also be a container action. The “change style”
25 operation 308 includes a primitive component action 312. “Primitive” denotes

1 that the action is the lowest level (or most decomposed) action available through
2 the action management facility. It should be noted, however, that lower level
3 actions may exist, such as individual method calls or microprocessor instructions.

4 The levels of granularity available are dictated by the individual action
5 management facilities of the various sources, including the various applications,
6 objects, and the computer systems. It should also be understood that one or more
7 of the component actions 304, 306, 308, 310, and 312 may be selected for an
8 action management operation. For example, the “change style D1” action 308
9 may be selected for an undo operation, which results in the undoing of its
10 component action 312, while leaving the other unrelated component actions 304,
11 306, and 310 of the container action 302 unchanged.

12 To contrast the hierarchy associated with action 302 with that of action 300,
13 consider execution of an action management operation on container action 300,
14 which has component actions 314, 316, 318, and 320. For example, undoing the
15 action 300 results in the undoing of the component actions 314, 316, 318, and 320,
16 and, necessarily, the component actions thereof (such as component actions 322,
17 324, and 326 of action 318). It should also be noted that a relationship between
18 action 320 and action 306 is indicated by dashed arrow 328. Therefore, in one
19 implementation, the action 306 is also undone.

20 FIG. 4 illustrates a user interface window 400 for an exemplary selective
21 action management facility. A system-wide aspect of the UI is shown, as denoted
22 by the various icons shown in each action block (e.g., a file system icon 402 in a
23 file system “Make Dir” action 404 as compared to a word processing icon 406 in a
24 word processing “Edit Doc” action 408).

1 Actions are displayed at a given level of granularity in a causal order (i.e.,
2 showing predecessor-successor relationships). One alternative ordering that may
3 be displayed is based on the chronological execution of each action within the
4 system, although other orders may also be employed. In addition, actions may be
5 filtered as to source, system level, and other characteristics, such that the number
6 and/or character of displayed actions may be limited.

7 The user may zoom into the container action 408 to display its component
8 actions 410, 412, 414, and 416 in zoom window 418. These component actions
9 410, 412, 414, and 416 are shown as actions performed in a word processing
10 document, as denoted by the document icon 420.

11 Various action management operation controls are show in the lower left
12 corner of the user interface window 400. An undo control 422 allows a user to
13 undo one or more selected actions. For example, the user may select the
14 action 432 and then activate the undo control 422 to trigger the “undoing” of the
15 action 432. As illustrated, actions 434 and 408 are dependent upon the action 432.
16 Therefore, actions 434 and 408 are also undone. The results of the undo operation
17 are persisted in association with each source.

18 An undo preview control 426 allows a user to preview a selected undo
19 operation, without persisting the results in association with each source. For
20 example, the results of the undo operation may be propagated to all related
21 actions, which may be reflected in changes to the display of the UI. In one
22 embodiment, if all related actions are not shown in the current display, the UI may
23 change to display the action information at a higher level so as to encompass all
24 changes to related actions. To persist the results of the undo preview operation, an
25 undo operation for the same selected actions may be activated.

1 A redo control 424 allows a user to redo one or more selected actions. To
2 redo consists of making active, and executing, an action that was previously
3 inactive, and similarly for all actions that depend (directly or indirectly) upon it.
4 For example, the user may select text in DocA, select the action 432, and then
5 activate the redo control 424 to trigger re-execution of the action 432, 434
6 and 408. When completed, the results of the redo operation are persisted in
7 association with each source.

8 A redo preview control 428 allows a user to preview a selected redo
9 operation, without persisting the results in association with each source. For
10 example, the results of the redo operation may be propagated to all related actions,
11 which may be reflected in changes to the display of the UI. In one embodiment, if
12 all related actions are not shown in the current display, the UI may change to
13 display the action information at a higher level so as to encompass all changes to
14 related actions. To persist the results of the redo preview operation, a redo
15 operation for the same selected actions may be activated.

16 Although not specifically illustrated in FIG. 4, another exemplary action
17 management operation is a “fix” operation, in which an action is modified or
18 replaced with a different action. For example, a user may “fix” action 432 by
19 selecting it and activating a “fix” operation through a provided control (not
20 shown). The resulting operations are generally source-dependent and action-
21 dependent. Nevertheless, in one “fix” example, a context menu may be displayed
22 offering valid “fix” options, such as inserting additional actions, changing the
23 action itself (e.g., from “Make Dir” to “Create File”), or changing the parameters
24 (e.g., “DirD”) of the action 432. Accordingly, in one implementation, the “fix”
25 operation causes the original action 432 to be undone and replaced with one or

1 more corrected actions (e.g., an action of the same action type as action 432 but
2 with a different directory name, a different action, etc.).

3 As mentioned above, multiple actions may be selected as the target of an
4 action management operation. In one implementation, actions may be ordered
5 chronologically, such that multiple actions can be selected in accordance with the
6 timing of their execution. As such, a user may select actions in such a way that all
7 actions back to a given action are undone (or “rolled back”). In addition, action
8 relationships (e.g., causal dependencies or atomicity) can result in the effective
9 selection of multiple actions based on the manual selection of a single action.

10 Action relationships can be determined automatically within the individual
11 action management facilities or they may be specified by the user through a
12 relationship control. An exemplary relationship control is shown as “parcel”
13 control 430, which can define a set of actions as “atomic” in nature, such that an
14 action management command applied to one action in the parcel must be
15 propagated to all other actions in the parcel. In contrast, a causal relationship may
16 be defined between two actions, such that undoing a predecessor action propagates
17 the undo action to the successor action, but an undo operation does not propagate
18 from successor to predecessor. Another example relationship is the “alternative”
19 whereby if one action is active, the other is inactive. Another class of exemplary
20 relationships includes time-based constraints. For example, a relationship between
21 two actions may require that an action management command applied to one of
22 the actions must be applied to the other within a specified time interval.

23 FIG. 5 illustrates operations 500 of an exemplary selective action
24 management process. A registration operation 502 registers a first computer-
25 related source with an action management module. Another registration

1 operation 504 registers a second computer-related source with the action
2 management module. A logging operation 506 logs the actions and associated
3 relationships of the first source (e.g., in an action log or by passing action
4 information to an action management module). A logging operation 508 logs the
5 actions and associated relationships of the first source.

6 A user interface operation 510 displays action information in an action
7 management user interface, such as the one shown in FIG. 4. A selection
8 operation 512 receives a selection of one or more actions and an action
9 management command (e.g., undo, redo preview, parcel, etc.). An identification
10 operation 514 identifies one or more actions related to the selected action(s) and
11 the sources corresponding to the selected actions and the related actions. In one
12 implementation, the identification operation 514 also identifies the distributed
13 action logs associated with each action. An action management operation 516
14 executes the action management command on the selected action(s) and
15 appropriate related actions.

16 To better understand how selective action management might be
17 implemented, consider some set of documents, such as DocA and DocB in FIG. 3.
18 Each action such as 314 or 316 is a programmatic operation that changes the state
19 of the associated document. For any given action type, the application may
20 register a “compensation” that reverses its effect. For example, a “type character”
21 action might be associated with a “delete character” compensation. As the user
22 submits actions and relations, the system executes active actions as dictated by
23 their relations and the state of the documents changes. For example, after
24 action 314 “Type D1”, the state of DocA has changed to incorporate D1. In one
25 implementation, the roll back of the single action 314 may be achieved by

1 applying the compensation “delete D1”. Rolling back a set of actions (e.g. 314
2 and 316) is achieved by applying the corresponding compensations in reverse
3 order (viz. compensate 316 then compensate 314). In an alternative
4 implementation, the system occasionally records the current state in persistent
5 state variables called “checkpoints”. Then, rolling back a set of actions is
6 achieved by returning the state to the nearest saved checkpoint, and either
7 compensating or re-executing actions as appropriate.

8 In support of a system-wide selective action management facility, a file
9 system can record and identify different checkpoints (also called versions) of files,
10 record logs, record dependency information between log records and between a
11 log record and a file checkpoint, etc.

12 In one implementation, a user interface to a system-wide selective action
13 management facility provides a high-level view of the system over time. A user
14 can zoom in and out to different levels of granularity. The user can view actions
15 associated with different sources (e.g., applications, sites, documents, objects,
16 processes, computer systems, system level, etc.) and relationships among the
17 actions. The user can also review previously executed actions that resulted in an
18 error and the consequences of undoing, fixing, and/or redoing the previously
19 executed actions.

20 The different levels of granularity can be controlled by various parameters,
21 including:

22 (1) The *source* of the actions – which application document, site,
23 user, object, system level (e.g., application level, kernel level,
24 method level, microprocessor level, etc.) is responsible for the
25 actions

- (2) The *orientation* of the actions – chronological relationships, causal relationships, etc.
- (3) A *filter* – which may be applied to refine the focus of the display. For example, a filter may focus on a specific time period, or actions that modify a particular portion of a document. In another example, an administrator may focus on time periods in which system configuration objects (e.g., a configuration file or registry) had been recently updated.

The scope of an action management operation is defined as the set of actions that are undone, fixed and redone by the action management command. The user is notified if the selected action management operation would have ramifications outside of the selected scope. The scope of an undo should include all the actions that depend on the action to be undone.

The scope of persisted action management operations (e.g., undo, fix, redo) is carefully controlled by the action management facility. For example, action management operations remain in compliance with security policies. Likewise, pathological conditions, such as a “domino effect” that could cause an extreme rollback of action, may be avoided. As such, an action management operation commanded with a given scope may be denied in any one or more of the following cases:

- a. The command specifies an undo operation, and an action in the scope is explicitly designated as “not undoable”.
- b. An action management operation commanded within the scope cannot be physically performed (e.g., the source is no longer available).

- 1 c. An action management operation commanded on an action within
- 2 the scope fails (e.g. if the user fixes an action, and re-executing one
- 3 of the dependent actions fails, then the whole undo-fix-redo
- 4 command is denied).
- 5 d. The user does not have authorization to perform the action
- 6 management operation within the scope.
- 7 e. An action that is not in the selected scope would have to be undone
- 8 or redone. (For example, given actions A, B, and C where B
- 9 depends on A, and B and C are in a parcel, if the user requests to
- 10 undo A, then both B and C would need to be undone, even through
- 11 C is not dependent upon A.)
- 12 f. Action information for the selected action or a related action is
- 13 missing (e.g., the action log is corrupted).

14 Action log size is readily controlled by limiting the allowed recovery period
15 (e.g., to one year). Older logs can be put into tertiary storage or removed
16 altogether. Another approach is to coarsen the available time granularity of action
17 management operations. For example, if the user is allowed to go back to any
18 specific point in time, the number of log entries is quite large. However, if the
19 user is only allowed to go to fixed points in time (e.g., the end of each business
20 day), all intermediate logs during the day may be erased. Time-limiting and
21 granularity control may be combined such that increasing coarse granularity exists
22 farther in the past. Furthermore, actions that precede an action that cannot be
23 undone can be pruned from the action log.

24 From time to time, both application-specific and system-wide action
25 management facilities may record the state of their related source(s) as

1 “checkpoints”. However, checkpoint data may also cause significant disk usage.
2 The greater number of checkpoints recorded, the quicker the action management
3 operation may be performed, but at the expense of disk utilization. As such, the
4 interval of checkpoints may be increased as the checkpoints get older (i.e., older
5 checkpoints at various intervals may be erased as time passes).

6 The exemplary hardware and operating environment of FIG. 6 for
7 implementing the invention includes a general purpose computing device in the
8 form of a computer 20, including a processing unit 21, a system memory 22, and a
9 system bus 23 that operatively couples various system components include the
10 system memory to the processing unit 21. There may be only one or there may be
11 more than one processing unit 21, such that the processor of computer 20
12 comprises a single central-processing unit (CPU), or a plurality of processing
13 units, commonly referred to as a parallel processing environment. The computer
14 20 may be a conventional computer, a distributed computer, or any other type of
15 computer; the invention is not so limited.

16 The system bus 23 may be any of several types of bus structures including a
17 memory bus or memory controller, a peripheral bus, a switched fabric, point-to-
18 point connections, and a local bus using any of a variety of bus architectures. The
19 system memory may also be referred to as simply the memory, and includes read
20 only memory (ROM) 24 and random access memory (RAM) 25. A basic
21 input/output system (BIOS) 26, containing the basic routines that help to transfer
22 information between elements within the computer 20, such as during start-up, is
23 stored in ROM 24. The computer 20 further includes a hard disk drive 27 for
24 reading from and writing to a hard disk, not shown, a magnetic disk drive 28 for
25 reading from or writing to a removable magnetic disk 29, and an optical disk drive

1 30 for reading from or writing to a removable optical disk 31 such as a CD ROM
2 or other optical media.

3 The hard disk drive 27, magnetic disk drive 28, and optical disk drive 30
4 are connected to the system bus 23 by a hard disk drive interface 32, a magnetic
5 disk drive interface 33, and an optical disk drive interface 34, respectively. The
6 drives and their associated computer-readable media provide nonvolatile storage
7 of computer-readable instructions, data structures, program modules and other
8 data for the computer 20. It should be appreciated by those skilled in the art that
9 any type of computer-readable media which can store data that is accessible by a
10 computer, such as magnetic cassettes, flash memory cards, digital video disks,
11 random access memories (RAMs), read only memories (ROMs), and the like, may
12 be used in the exemplary operating environment.

13 A number of program modules may be stored on the hard disk, magnetic
14 disk 29, optical disk 31, ROM 24, or RAM 25, including an operating system 35,
15 one or more application programs 36, other program modules 37, and program
16 data 38. A user may enter commands and information into the personal computer
17 20 through input devices such as a keyboard 40 and pointing device 42. Other
18 input devices (not shown) may include a microphone, joystick, game pad, satellite
19 dish, scanner, or the like. These and other input devices are often connected to the
20 processing unit 21 through a serial port interface 46 that is coupled to the system
21 bus, but may be connected by other interfaces, such as a parallel port, game port,
22 or a universal serial bus (USB). A monitor 47 or other type of display device is
23 also connected to the system bus 23 via an interface, such as a video adapter 48.
24 In addition to the monitor, computers typically include other peripheral output
25 devices (not shown), such as speakers and printers.

1 The computer 20 may operate in a networked environment using logical
2 connections to one or more remote computers, such as remote computer 49. These
3 logical connections are achieved by a communication device coupled to or a part
4 of the computer 20; the invention is not limited to a particular type of
5 communications device. The remote computer 49 may be another computer, a
6 server, a router, a network PC, a client, a peer device or other common network
7 node, and typically includes many or all of the elements described above relative
8 to the computer 20, although only a memory storage device 50 has been illustrated
9 in FIG. 6. The logical connections depicted in FIG. 6 include a local-area network
10 (LAN) 51 and a wide-area network (WAN) 52. Such networking environments
11 are commonplace in office networks, enterprise-wide computer networks, intranets
12 and the Internet, which are all types of networks.

13 When used in a LAN-networking environment, the computer 20 is
14 connected to the local network 51 through a network interface or adapter 53,
15 which is one type of communications device. When used in a WAN-networking
16 environment, the computer 20 typically includes a modem 54, a network adapter, a
17 type of communications device, or any other type of communications device for
18 establishing communications over the wide area network 52. The modem 54,
19 which may be internal or external, is connected to the system bus 23 via the serial
20 port interface 46. In a networked environment, program modules depicted relative
21 to the personal computer 20, or portions thereof, may be stored in the remote
22 memory storage device. It is appreciated that the network connections shown are
23 exemplary and other means of and communications devices for establishing a
24 communications link between the computers may be used.

1 In an exemplary implementation, an action management module, an action
2 management UI, various applications, and other modules may be incorporated as
3 part of the operating system 35, application programs 36, or other program
4 modules 37. Action logs and other data may be stored as program data 38.

5 The embodiments of the invention described herein are implemented as
6 logical steps in one or more computer systems. The logical operations of the
7 present invention are implemented (1) as a sequence of processor-implemented
8 steps executing in one or more computer systems and (2) as interconnected
9 machine modules within one or more computer systems. The implementation is a
10 matter of choice, dependent on the performance requirements of the computer
11 system implementing the invention. Accordingly, the logical operations making
12 up the embodiments of the invention described herein are referred to variously as
13 operations, steps, objects, or modules.

14 The above specification, examples and data provide a complete description
15 of the structure and use of exemplary embodiments of the invention. Since many
16 embodiments of the invention can be made without departing from the spirit and
17 scope of the invention, the invention resides in the claims hereinafter appended.